BETA BASIC NEWSLETTER No. 1

Welcome to the first issue! (This is its second printing. A few errors - mentioned in Newsletter No. 2 - have been corrected.) The Newsletter is intended to provide hints, examples and advice on Beta Basic programming. We are keen to include contributions from readers - we have already collected quite a few. Each issue will also cover questions asked by users. (Previous correspondents will know we answer letters anyway - but this way points of general interest can reach a wider readership.) As new Betasoft products become available, we will inform you, without much "marketing blather- speak". (You know, all that "UNBELIEVABLE INCREDIBLE MEGA WONDER PROGRAM Stuff.)

I suppose I had better introduce myself, since this is the first issue. Your Editor (i.e. ME) is Dr. Andrew Wright, former.: medical research scientist turned programmer, generally known as Andy. I wrote the Beta Basic program and manuals for Betasoft, so I should be able to handle most readers' queries.

You may wonder what your fellow Beta Basic users are like. Judging from letters and personal contacts, most users are over 20, which makes them ancient on the Spectrum scene. Many are professionals - photographers, engineers, programmers, medics, etc. Some are retired, some are students, some are unemployed. More than you might expect live outside the U.K. In general, I would say they are the "crème de la crème" of Spectrum users - so lets have some nice contributions!

Because long programs will fill up an issue quite quickly, the programming contributions we like best are fairly short. To be as generally useful as possible, these should often be in the form of procedures, with an example program that calls them. Contributions from Betasoft will usually follow this pattern. We usually give listings in LIST FORMAT 2, and do not use many line numbers. (When writing a procedure, we often use quite frequent line numbers, and later unite all the lines into one with JOIN.)

Procedures effectively provide new commands, and this is relevant to the many letters we get suggesting future enhancements. Because memory is so tight, any new commands for a future Beta Basic should be either impossible or very slow to achieve with a procedure. For example, a user suggested (among other ideas) a command to store large sections of the screen. This has low priority, as it can be done from Basic almost as fast as if it were written in machine code - see the procedures GRAB and PUT in this issue. The same user also suggested a modification to PLOT to allow rotation of the characters. This has a much higher priority, since it can be done with a short and fast code program, but is almost impossible from Basic. Other common suggestions sound simple, but are very complex to implement, particularly since the Spectrum ROM is hard to get round. One example is providing long string names or multi-line functions - it sounds easy, but any alterations to expression evaluation apart from via DEF FN (as we have done in line zero) are difficult. The reasons are too technical to bother with unless readers are interested. Other ideas are very easy e.g. MEMORY$, AND, OR and DPEEK take less than 20 bytes each.

Any major expansion of Beta Basic will probably require either a 128K Spectrum, or chopping up of the program into specialised chunks.

****
PROC BINAR - binary to decimal conversion.

We already have binary to decimal conversion, with BIN, so what do we need binary for? Well, *binar allows you to use a variable* binary (string) input, which BIN doesn't. binar is pretty sneaky - it takes the string which is provided as the first parameter e.g. "1010" , and then uses KEYIN (a subtle command!) to effectively "type":

```
LET binary=BIN 1010
```

The value of "binary" is passed to whatever variable was specified by the second parameter - in the example below (lines 10-60) this is "result".

```
10 DO
20     INPUT b$
30 EXIT IF b$=""
40     binar b$,answer
50     PRINT answer
60 LOOP
70
100    DEF PROC binar n$,REF binary
         KEYIN " LET binary=BIN"+n$
       END PROC
```

****
PROC OVERWINDOW - superimposed windows.

Some versions of the WINDOW command (usually on larger machines) allow windows to be superimposed over what is on the screen and then removed later to reveal the material underneath once more We were reluctant to provide this as an automatic facility (it eats memory). However, you can still achieve this. The procedure below sets up a superimposed window in the middle of the screen if overwindow 1 is used, and removes it if overwindow 0 is used. It has two "niggles": it uses a non-local z$ to store the area which is over-written, and it uses WINDOW 10. You will have to watch this when you use it in your own programs. If you want the "overwindow" somewhere else, change x and y, but remember that attribute problems are minimised if the window covers whole character squares. X should be divisible by e, and so should (y+1).

```
10 PRINT STRING$(90,"testing")
20 overwindow 1
30 PRINT "HELLO THERE," "'GENTLE READERS!": REM or CAT
40 PAUSE 100
50 overwindow 0

200 DEF PROC overwindow yes
      LOCAL x,y,w,h
      LET x=64,y=127,w=126,h=96
      IF yes THEN
        GET z$, x, y, w/8, h/8 ;1
        WINDOW 10,x,y,w,h
        WINDOW 10
        PAPER 6
        INK 1
        CLS
      ELSE
        PLOT x,y;z$
        WINDOW 0
201 END PROC
```

Note: The END PROC must be on a separate line.

****

TIMING PROCEDURES: ztime (zero timer), ptime (print timer) and gtime (get timer value).

We use these procedures to time different ways of programming things - for example, in writing PROC bplot (see later) we wanted to know if it was faster to plot 4 pixels at once using 4, wanted or using a UDG. They access the system variable FRAMES and are independent of CLOCK. The greater accuracy compared with CLOCK, and the fact that a numeric result is directly available (unlike TIMES) are often convenient. Periods of up to 21 minutes can be timed, accurate to 1/50th. of a second.

```
10 ztime
20 FOR n=1 TO 10: NEXT n: REM thing to time
30 ptime
```

or:

```
30 gtime t: IF t>2.5 THEN PRINT "That's a long time!"
```

```
210 DEF PROC ztime
      DPOKE 23672,0
    END PROC
```

```
220 DEF PROC ptime
      PRINT USING "###.##";DPEEK(23672)/50-0.02
    END PROC
```

```
230 DEF PROC gtime REF secs
      LET secs=DPEEK(23672)/50-0.02
    END PROC
```

****

PROC UDG - defining User-Defined Graphics.

Since it turns out that it is faster to plot one UDG than 4 pixels, we wanted a UDG-defining procedure. (This is used by the next 2 procedures.) It illustrates the use of a DATA list instead of the 9 parameters you would otherwise need. Note: due to a slightly user-unfriendly (or is it unuserfriendly?) piece of code, it is important that there be no space between "udg" and the DATA keyword (although if you type space-d-a-t-a all is well because all five characters are converted to the keyword.)

```
240 DEF PROC udg DATA
      LOCAL a$,n,x
      READ a$
      FOR n=0 TO 7
        READ x
        POKE USR a$+n,x
      NEXT n
    END PROC
```

Below, the procedure is called to define graphic "a", using the required 8 numbers:

```
100 udg "a",BIN 11000000,BIN 11000000,0,0,0,0,0,0
```

That graphic is then used by the procedure BPLOT (R.T.O.).

****

PROC BPLOT - large pixel PLOT.

Note: This procedure assumes graphic-A has been defined as a 2*2 pixel dot, as in the UDG procedure example. KEYWORDS is used to turn on user-defined graphics temporarily (ALTER should be entered as graphic-A). OVER 2 is used so that only the INK pixels of the graphic have any effect.

```
110 DEF PROC bplot x,y
      KEYWORDS 0
      PLOT OVER 2;x,y;" ALTER "
      KEYWORDS 1
    END PROC
```

To draw a line with large pixels, you can put them at 2-pixel intervals, or re-define the graphics scale with XRG and YRG.

****

PROC sketch - sketchpad.

This procedure uses PROC bplot above. It is fairly primitive (only 4 directions are recognised) but this should just stimulate you to improve it!

```
160 DEF PROC sketch
      LOCAL x,y,d,c$
      LET x=128,y=88,d=2
      LET c$="8576"
      DO
        ON INSTRING(i,c$,INKEY$)
        LET x=x+d
        LET x=x-d
        LET y=y+d
        LET y=y-d
170 bplot x,y
      LOOP UNTIL INKEY$=" "
    END PROC
```

As written, the procedure responds to keys 5,6,7 and 6, which makes sense if you have a Spectrum (not Spectrum plus) keyboard. Movement control is by INSTRING searching a string of control keys for a match with INKEY$ - the position value is then used by ON to control which statement is executed. The use of INSTRING to interpret user input in this way is a useful technique, especially if there are lots of possibilities. (How could you cope with upper case/lower case?)

The string of control keys (c$) could be "rlud" (for Right, Left, Up, Down) or CHR$ B+CHR$ 9+CHR$11+CHR$ 10 (for Spectrum Plus cursor keys).

ON requires a new line to terminate its action, so this procedure must be written on two lines.

To use the procedure with pixels of different sizes, change the value of "d", and alter the udg used. (You can just use d=1 and the ordinary PLOT command, of course.)

You may want to use this procedure with the next one - SHRINK.

****

PROC shrink - shrink the screen to a quarter size.

The November 85 "Your Spectrum" had a MegaBasic shrink procedure which looked so awful I decided to write one! Theirs uses POINT and PLOT and nested FOR-NEXT loops, but we should be able to do it a bit easier than that:

```
230 DEF PROC shrink
      LOCAL a$
      GET a$,0,175,32,22;1
      PLOT CSIZE 4,4;0,175;a$
END PROC
```

The procedure GETS the whole screen, and then PLOTS it back at half size in the top left corner. You can "soup it up" to your own taste. It works quite nicely on game loading screens, but with some attribute problems in places. Also, lines one pixel thick may vanish. You might want to use PROC bplot to design graphics, at a convenient size, and than use PROC shrink to reduce everything to normal pixel size.

****

THE WORLD'S MOST USELESS PROGRAM.

Betasoft's entry:

```
100 DEF PROC off
      CLS
      BORDER 7
      ALTER TO PAPER 0
      PAUSE 70
      ALTER TO PAPER 7
      PRINT #1;"C 1982 Sinclair Research Ltd"
      PAUSE 150
      PRINT CSIZE 16;"JUST A JOKE!"
      BEEP 1,-35
END PROC
```

Malcolm Goodman's entry (17 Brookhill Avenue, Leeds, LS17 BOA) (particularly useless since it doesn't use Beta Basic!):

```
10 RANDOMIZE USR 3884
20 PRINT AT 11,12;"SHAN'T!"
30 BEEP .1,-10: PAUSE 30: CLS
40 GO TO 10
```

Malcolm would like to correspond with users in his area who "know a bit of Basic, a lot less machine code, but want to find out what the Spectrum can do". He is particularly interested in protection techniques. (Presumably he wants to protect the programs he has written...)

****

NOWOTNIK PUZZLE

This contribution from J.M. Crawford of Harrogate is a version of a puzzle by David Nowotnik. He says: "The programming is original, although the idea is not. The QL version (ZX Computing, June/July 85) ran to 234 lines and 3 pages." This version lacks some unimportant features of the QL version (such as checking for correct completion) but it works very well. Apart from being listed in LIST FORMAT 1, and having a SAVE routine removed the program is as we received it. It will work with Beta Basic 1.8 or 3.0. Once the program is run, press key 9 to scramble the puzzle, then use the keys 1-8 to get back to the initial position It is surprisingly difficult! Press key 0 to RUN again.

```
30 BRIGHT 1
   BORDER 1
   PAPER 1
   INK 7
   CLS
   LET A ="   "
45 PRINT ,T 1,10;"<5";AT 1,20;"6>";AT 20,10;"<7";AT 20,20;"8>
46 PRINT AT 5,6;"^";AT 6,6;"1";AT 5,25;"^";AT 6,25;"3"
47 PRINT AT 15,6;"2";i':T 15,25;"4";AT 16,6;"v";AT 16,25;"v"
50 FOR N=1 TO 8
       PRINT AT 2+N,8; PAPER 2;A$; PAPER 6;A$;AT 10+N,8; PAPER 4;
       A$;PAPER 5; A$
          NEXT n
       55 PLOT 59,20
          DRAW 0,136
          DRAW 136,0
          DRAW 0,-136
          DRAW -136,0
       60 GET A
          IF A THEN GO SUB 200
              GO TO 60
           ELSE RUN
       100 ROLL 3,8;64,144;8,128
              RETURN
       110 ROLL 2,8;64,152;8,128
              RETURN
       120 ROLL ,8;128,144;8,128
              RETURN
       130 ROLL 2,8;128,152;8,128
              RETURN
       140 ROLL 1,8;64,144;16,64
              RETURN
       150 ROLL 4,8;64,144;16,64
              RETURN
       160 ROLL 1,8;64,80;16,64
              RETURN
       170 ROLL 4,8;64,80;16,64
              RETURN
       180 FOR M=1 TO 5
              LET A=(8*RND)+1
              GO SUB 200
           NEXT M
           RETURN
       200 FOR N=1 TO 4
              GO SUB ON A;100,110,120,10,140,150,160,170,180
           NEXT N
           RETURN
```

****

PROCs GRAD and PUT - screen handling.

Dave Trebilcock (Cheshire) suggested implementing an improved POKE command to avoid having to work out screen addresses when storing and replacing parts of the screen with MEMORY$ and POKE. The procedures GRAD and PUT go some way towards meeting his request, but they could easily be improved. As written, GRAD can store any third of the screen (1 being the top, 2 the middle, and 3 the lower third) with its attributes, to a specified string. The top third is the default. PUT can copy a specified string onto any third of the screen (default is the top third again). Line 10 prints to the top third, line 20 uses grab to store that third, and line 40 copies the string back to all three thirds. The method is a lot less flexible than the GET command, but it is rather faster, and better suited to animation.

```
10 FOR n=1 TO 64
       PRINT CSIZE 16; INK RNDM(6);CHR$ (RNDM(25)+65);
   NEXT n
20 grab a$: REM or grab a$,1 (grab a$,2 for middle,etc.)
30 CLS
40 FOR n=1 TO 3
       PAUSE 50
       put a$,n
   NEXT n
50 PAUSE 0
60
100 DEF PROC grab REF a$,third
110    DEFAULT third=1
115    LET third=third-1
120    LET a$=MEMORY$()(16384+third*2048 TO 18431+third*2048)+
       MEMORY$()(22528+third*256 TO 22783+third*256)
130 END PROC
140
150 DEF PROC put a$,third
160    DEFAULT third=1
165    LET third=third-1
170    POKE 16384+third*2048,a$( TO 2048)
       POKE 22528+third*256,a$(2049 TO 2304)
180 END PROC
```

****

READERS LETTERS.

Sirs,

I don't know how many of your "boffins" are knocking at, or have knocked on 70 years (age) but many of your customers must have. Now when we went to school even radio was mentioned in whispers, and programs like yours were not even in science fiction books (I read them all) but colours were about and we knew that red in half light looked like black; so that black ink on red soon became black on black. I'm not suggesting that 70 year olds live in half light but our eyes are not sparkling any more. So please can we have some paler coloured paper.
Yours Truly,
T. Cawley, Manchester

The first version of Beta Basic had a black-on-white manual, but the program itself was protected. This obviously caused problems for backing-up, Microdrive copies, program modifications, etc, so the protection was removed for version 1.8. On the other hand, visits to computer clubs had shown an alarming level of

pirating, including photocopied documentation. The current crop or bankruptcies should show you that making a profit in this business isn't that easy, so some protection was obviously a good idea. Since Beta Basic is difficult to use without a manual, we looked for colours that were hard to photocopy. Red looks black to a photocopier, so black on red is uncopiable. Since red also looks black to the rod cells used by the human eye at low light levels, the manual must be well-lit to be readable. He investigated many paper colours, and also tried blue ink (very pale to a photocopier) but the current combination was the only satisfactory one. Ne are very sorry that some customers have problems, but we feel we cannot discard protection altogether.

I've read a lot of old SF too! It seems modern developments were anticipated - trends were extrapolated to predict very good slide rules, very fast punched card readers, and very small valves (those glass things).

Can you tell me how to disable the improved BREAK command? Dr. D.E. Hoffler, Leicester.

Easy - just POKE 63150,24

I wish to produce copies of Beta Basic programs I have written. Please send me details of how to alter Beta Basic's code to a "RUN only" version. Thank you for a very good program. The next must be a Beta Basic compiler!
Yours faithfully, Alf Roksvag, Norway

You can alter Beta Basic's CODE with the following single line:

```
POKE 64218,0: POKE 64219,0: POKE 64220,0: POKE 64861,201
```

This will allow direct commands (such as RUN) to be obeyed, but programs can no longer be edited or written.

Unfortunately, a good compiler might take a man-year to write, and it could be rather large. Considering it could only be sold to BB users, the sales probably would not be worth the programming effort. In any case, the approach of adding new commands can often be faster - for example, SORT, INSTRING, ROLL, POKE string, and ALTER are much faster than a compiled version of ZX Basic to do the same job!

The CSIZE option is extremely useful but when used with PRINT it can be a bit confusing. Because it redefines the SPECTRUM screen it can be difficult to calculate the correct AT position. Also it denies the chance of printing at some places on the screen e.g. CSIZE 16,16 eliminates all odd print positions. Of course PD's PLOT allows these positions to be used but the calculation of the correct position is even harder. The following short procedure allows the use of PLOT to simulate PRINT AT but combining the ability to print at any normal SPECTRUM character square.

Dave Trebilcock, Cheshire

```
9000 DEF PROC pplot w,h,L,c,z$
9010   REM w=width, h=height, L=line no., c=column no.
9020   DEFAULT w=8,h=8,L-0,c=0
9020   PLOT CSIZE w,h;c*8,(22-L)*8-1;z$
9040 END PROC
```

This has the advantage or working with any "global" CSIZE. However, if you just want to place a few strings on the screen at different CSIZEs when you are in CSIZE O (with the normal AT system) you might prefer to do this:

```
PRINT AT L,c; CSIZE w,h;z$
```

The print position Nil! be set to L,c on the normal AT system before the CSIZE is encountered. Then the string will be printed at that position in the specified CSIZE. On the other hand:

```
PRINT CSIZE w,h; AT L,c;z$
```

will use the AT system appropriate to the specified CSIZE.

Dave also added that he'd like to see a do-everything BB demo cassette, which seems like a good idea.

.....I am using KEYIN to collect UDG values into strings in my program, so that I can delete great long DATA lists containing UDG values_ I wanted to write:

```
KEYIN "100 POKE 65368,"""+MEMORY$() (65368 TO 65535)+""""
```

(*That should add the line below to the program - Ed.*)

```
100 POKE 65368,"(string from UDG area)"
```

After finding the restriction on MEMORY$ and changing 65535 to 65532 it worked O.K. as long as:

1. there were no CHR$ 34's in the UDGs.
2. there were no CHR$ 13's in the UDGs.

1 is understandable, but 2 1 don't understand. Is this a feature or a bug? Is there a better way to do this so that there are no restrictions?
Yours truly,
Henry Cosh (Berks.)

As you say, not being able to include CHR$ 34's (") is understandable - you could not enter such a line from the keyboard, and that is what KEYIN imitates. Less obviously, you cannot include CHR$ 13, which is the end-of-line marker.

You could instead store the UDG data string to a REM statement. First create one of the correct length:

```
KEYIN "100 REM"+STRING$(165,"X")
```

Make the line immediately above:

```
99 LET x=DPEEK(23637)+5: POKE x,MEMORY$()(65368 TO 65532)
```

This uses the system variable NXTLIN to find the address of the first byte after REM and POKES your current UDGs there. To replace the data (when you have re-loaded the program, presumably) the line should be:

```
99 LET x=DPEEK(23637)+5: POKE 65368,MEMORY$()(x TO x+164)
```

One advantage is that the data is safe from RUN, but the REM statement can mess up the listing. (Put it at the end of the program?) you might prefer instead to assign the UDG area of MEMORY$ to a string, and save the program so that it auto-runs (actually, "auto-running" programs really "auto-GO TO") and POKEs the string back into the UDG area.

Dear Sirs,

Is it possible to prevent keywords being shown as such in REM statements? Something like

20 REM Procedure TO DRAW a CIRCLE AND a LINE

looks a bit grotty. Typing

20 REM Procedure to draw a circle and a line

leads to the same result.
Yours faithfully, G.A. Duller, Devon

I should have anticipated the REM problem and "turned off" keyword creation. However, you can easily do this yourself by typing KEYWORDS 2, then the REM line, then KEYWORDS 3 (KEYWORDS is Graphic-8). Some early versions of BB 3.0 incorrectly made keywords out of odd combinations of characters often containing "(" and ">". The version number of your cassette can be found by PRINT PEEK 47272 - the keyword creation bug was fixed in version . If you have an earlier version, you can exchange it at no charge.

Dear Sirs,

Have you any further news about the use of the Kempston E Centronics printer interface unit with Beta Basic? Some time ago you told me that Kempston were not at all helpful, and that some versions of their interface could be used but others not.
Yours faithfully,
L.W. Dewhurst, Leamington Spa

All sorted out now - see below. Kempston changed their technical queries chappie and the lack of response was due to our letters being lost in the re-shuffle, they say_ They put us onto the engineer who designed the interface - I heard his name with a groan, since he also designed the Euroelectronics interface and had already proved unhelpful. Host things I wanted to know he told me I didn't need to know - when I was damn sure I did! He seemed to think we wanted to pirate his firmware. Promised information on the kempston operating firmware never arrived, but I managed to work it out for myself.

The early Kempston E's worked with Beta Basic but not Masterfile. The later ones require a small alteration to BB's CODE to list the new keywords correctly. Load Beta Basic, then enter and RUN the program below. You can now SAVE the modified program using line I of BB's Basic loader. Note: If you use COPY: REM... after loading Beta Basic, you will have to re-initialise the program by entering RANDOMIZE USR 53419.

```
5 FOR n=58426 TO 58435
6 READ x: POKE n,x: NEXT n
7 DATA 35,24,9,87,219,251,243,195,146,15
8 POKE 61081,61: POKE 61082,228
9 POKE 60921,1: REM omit this LINE FOR BB 1.8
```

While we are on this subject, here is the "patch" for another popular printer interface:

ZX LPRINT III PRINTER INTERFACE

1.LOAD "Beta Basic" (both parts) and then type in:

POKE 58432,24: POKE 61081,0: POKE 61082,91

You can also POKE 60921,1 to send all control codes to the interface unmodified.

2.MERGE the Basic part of Beta Basic to get a copy of lines 1 and 2.

3.Edit line 2 of Beta Basic's loader to:

.....CLS : LPRINT: RANDOMIZE USR 58419.....

You should also include any setting of graphics, auto line feed, etc. just after the LPRINT.

4.RUN to save the modified program to tape (or edit lines 1 and 2 to save to Microdrive).

Dear Sir,

Any Microdrive user will have thankfully put their Beta Basic 3.0 cassette away on a remote shelf having loaded it into a cartridge; only to be retrieved in an emergency. BUT there comes the moment when having started some programming and made some progress with same, one realises that Beta basic would have been a great help. No good trying to LOAD BB because this will delete your work, nor will it MERGE from cartridge. The solution, apart. from finding the cassette once more and merging from that, is to put a second program onto the cartridge: right now! Do this. LOAD Beta Basic (or "run") from cartridge. Then MERGE "run". Alter line 1 by changing "run" (or what have you) to "bb". Delete LINE 2 from the SAVE statement. Delete the SAVE CODE statement. Then RUN.

Now, when you need it you can MERGE *"m";1;"bb" and then do a GOTO 2. Incidentally, this will cause your program to be run also. If this is not to your liking, add a STOP to the end of line 2 before doing the SAVE of "bb". This will result in a "Nonsense in BASIC" after your GOTO 2, but it can be ignored. Your program will be intact and Beta Basic will be resident.

Yours faithfully,
S. Charles Buszard, Chorleywood

Thanks for the contribution!

Dear Sirs,

Maybe you might give us the means to utilise and relocate some commands and functions to be incorporated individually in our own programs for personal use, i.e. SORT, ALTER, DELETE, RENUM, PROCEDURE COMMANDS, etc.
Yours faithfully,
E. Aloskofis, Athens

A common request! I wish it was possible, but there are a lot of problems. To understand any new command, a fair amount of code and various subroutines are required. The code is not re-locatable (it could have been, but at the cost of being considerably longer and slower) so any particular command or combination of commands needs some rewriting, A huge program with lots of data on jumps and calls could be written to do all this automatically, but it would be as much fun to write as a telephone ,directory, and would take almost as long. As soon as this pesky newsletter is finished, I will resume work on various specialised versions of BB. The first will be shortened by the omission of "toolkit" commands, and the second will be biased towards data handling.

Dear Dr. Wright,

Praise for program and its author deleted to save space! Ed.)
First, where could I POKE in data for the Swedish characters instead of the square brackets etc. in the "4 bit" character set? Rather naively, I suspect, I scanned your code for a character set somewhere, but obviously it is generated in a more devious way. Second, I noticed that LIST DATA etc. will not list variables beginning with the Swedish characters, which are allowed in the Scandinavian ROM. Would it be possible to correct this?
Rest Regards,
Joachim Smith, Sweden

Perhaps I should mention that many of the letters published here will have bits cut out to save space, leaving sections of general interest. (If the deleted sections contain questions, the writer will have them answered directly. The same goes for any query that looks urgent — e.g. printer problems.)

There is a visible small character set in BB, Mr. Smith, but it is easy to miss. Each pair of characters, from "space" to the copyright symbol, take 7 bytes, starting at 51271. The most significant 4 bits hold the even—numbered characters, and the least significant 4 bits hold the odd characters. An extra blank line is generated at the top of the characters when they are printed to make them S pixels tall. The following program shows the small character set — press any key to continue. CSIZE is Graphic shift—8.

```
10 KEYWORDS 0
   POKE 62865," CSIZE "
   POKE 62869,
   FOR n=51291 TO 51626 STEP 7
      PRINT AT 0,0;
      FOR b=0 TO 6
            LET a$=BIN$(PEEK (n+b))
            PRINT a$( TO 4);" ";a$(5 TO )
      NEXT b
   PAUSE 0
   NEXT n
```

Perhaps someone will send us a suitable character editor? You could use the BB functions AND and OR to avoid designing two characters at once.

Most readers will not know that there is a Scandinavian version of the Spectrum ROM with different characters sets, better memory check on switch-on, faster GOTO's and GOSUB's, and all bugs I know off removed (even the one that causes crashes when you load a big program from Microdrive, which I didn't understand before Mr. Smith sent me a corrected ROM copy). This ROM also allows a wider range of characters to act as variable names. If you POKE 49269,31 a wider range of variables can be displayed. With the ordinary ROM, the only extra variable you will see is f$ (curly bracket string), which is used by BB as a local variable store. Try LIST DATA inside a procedure.

****

PROC KEYPRINT - input routine with checks.

This contribution is from Tim Layton (Bristol). I have modified his listing format slightly by inserting extra colons and CHR$ 15's in some places. It struck me as an interesting (and possibly instructive) exercise to try and re-write Tim's useful procedure in a similar form, but without the use of GOTO. Therefore, the same procedure is given twice, as received, and as re-written. The first form may seem more intelligible to those used to standard Basic; the second would be preferred by "structure" fans.

```
   99  REM example program using "keyprint" procedure.
  100  CLS
       PRINT ""WHAT IS YOUR NAME?" ""(TYPE NAME THEN PRESS ENTER)"
       ' 'NAME: ";
  110  keyprint n$,20
  120  PRINT ''" REM n$= ";n$
  130  STOP

 9099  REM Original. This procedure prints a string from
       the current print position as it is being entered from the
       keyboard, Z passes a string by reference. DELETE is allowed
       & characters with codes between 32 and 127. The string is
       ended by pressing ENTER. The max string length L has a
       DEFAULT of 18.
 9100  DEF PROC keyprint REF k$,L
 9110   DEFAULT L=18
        REM L=max string length
 9120   LOCAL a$
 9130   BEEP .1,0 LET k$=""
 9140   PRINT CHR$ 95;CHR$ 8;
        GET a$
        IF (a$<CHR$ 32 OR a$>CHR$ 127)
        AND a$<>CHR$ 12
        AND a$<>CHR$ 13 THEN
          BEEP .1,0
          GO TO 9140
 9150   IF LEN k$=L OR a$=CHR$ 13 THEN
          PRINT " ";CHR$ 8;
          GO TO 9180
 9160   LET k$=k$+a$
        IF a$=CHR$ 12 THEN
             IF LEN k$>l THEN
                    PRINT " ";CHR$ 8;
                    LET kS=k$( TO LEN k$-2)
             ELSE GO TO 9130
 9170    PRINT a$;
        GO TO 9140
 9180 END PROC
```

```
 8099 REM re-written to avoid GO TOs.
 8100 DEF PROC keyprint REF k$,L
           DEFAULT L=18
           LOCAL
           LET k$=""
           REM L-max string length
 8110 DO
           PRINT CHR$ 95;CHR$ 8;
           DO
            GET a$
           EXIT IF
           (a$>CHR$ 31 AND a$<CHR$ 128)
           OR a$=CHR$ 12 OR a$=CHR$ 13
            BEEEP .1,0
           LOOP
 8120 EXIT IF LEN k$=L OR a$=CHR$ 13
 8130       IF a$<>CHR$12 THEN
            LET k$=k$+a$
            PRINT a$;
           ELSE
            IF LEN k$>0 THEN
                 PRINT " ";CHR$ 8;CHR$ 8;
                 LET k$=k$( TO LEN k$-1)
 8140 LOOP
 8150 END PROC
```

****

LAST WORD

Well, that seems to be all we've got time for this issue, even though there are some deserving contributions still unused. We hope this newsletter has given you plenty of ideas, some of them, and that you will be inspired to produce something interesting yourself. Until the next issue, Bye!

Our thanks to David Nowotnik for permission to use the basic concept of his puzzle.

Pages 15 & 16 Were blank

Scanned, Typed, OCR-ed, and PDF by

Steve Parry-Thomas 25th July 2004.

This PDF was created to preserve this
Newsletter for the future.

For all ZX Spectrum, Beta Basic
And www.worldofspectrum.org users

(PDF for Michael & Joshua)